AD-A071 140 ROCHESTER UNIV NY DEPT OF COMPUTER SCIENCE F/G 14/5
CONSTRAINT NETWORKS: MODELING AND INFERRING OBJECT LOCATIONS BY--ETC(U)
AUG 78 D M RUSSELL N00014-78-C-0164
UNCLASSIFIED TR-38 NL

1 OF 1
AD
A071140

END
DATE
FILMED
8 --79
DDC

12

LEVEL II

電腦科學

LEVEL II

rochester

Department of Computer Science
University of Rochester
Rochester, New York 14627

Accession For	
NTIS GRA&I	✓
DDC TAB	
Unannounced	
Justification Per the on file	
By	
Distribution/	
Availability Codes	
Dist.	Avail and/or special
A	

14 TR-38

6

Constraint Networks: Modeling and Inferring Object Locations by Constraints

10 Daniel M. Russell
Computer Science Department
The University of Rochester
Rochester, NY 14627

TR38

11 August 1978

12 30p

Abstract

Relationships between objects in the real world are constrained by many physical and functional considerations. This paper presents a formalism called Constraint Networks which allows such constraints to be represented and used to make inferences about object locations in images. Constraint Networks are used in a system which accepts information about geometric relationships between structures in images and then uses these constraints to guide search for these structures. The system has been used successfully to infer rib positions in a chest X-ray and to locate aeration tanks and new construction sites in aerial images.

-A-

The preparation of this paper was supported in part by the Defense Advanced ~~Research Projects~~ Agency, monitored by the ONR under Contract No. N00014-78-C-0164

15

410 386

[illegible]

I. Introduction

Consider the following situation: You are flying at 3000 feet somewhere close to home and wish to find your neighbor's house; you might find yourself examining the scene below in the following fashion: "Well, I know I live close to the river. Upstream from the end of my street is a park, so if I find the park I can find my house. And since my neighbor lives north of my house I can just look a little north of my house...". This impromptu strategy for neighbor's-house-finding can be viewed as applying a successive series of constraints to the aerial image, thus removing areas of the scene from further consideration. "My house is close to a river and down from the park" and "My neighbor's house is close to mine" are both facts which can be used to limit the search space for a feature to a reasonable size. The first constrains the sub-image to be scrutinized to that part of the total image which is both close to the river and at a particular orientation to the river and park, while the second implies a constrained sub-image which is close to your house. This process of successive reduction of the search space by repeated limitation of the space is extremely useful in approaching a computer vision task. Currently, many systems simply use the technology approach to vision; i.e. apply an operator over the whole of an image, and then use the results from this massive "search" for further analysis. If instead we could limit - or, as we will say, constrain - our focus of attention to a much smaller area by the intelligent use of some *facts and* inferences drawn from them, we could then make considerable savings in an analysis of any scene.

Our goal, then, is to maximize the use of facts we already know about a given scene to tell us where to look for objects we are interested in. Further, we would also like to be able to use any information that can be inferred during the analysis as soon as the inference can be made. To attain these goals, we introduce a formalism called Constraint Networks, which is useful in limiting our search of an image for a particular instance of an object.

II. Constraint Networks

Constraint Networks were originally outlined in [Ballard, et al.]. In this paper, I have extended and refined that notion. A Constraint Network (CN) models a real world object's expected location in an image by describing its relationships to other objects already located in the image. Each of these descriptions is actually a constraint on the object's location in the image. For instance, a dockyard is usually found adjacent to the water's edge and in or near a harbor. This statement tells us two characteristics of real dockyards: 1) dockyards are adjacent to the coastline; and 2) dockyards are in or near harbors. Both statements constrain the dockyard's possible location by specifying where it would be with respect to the coastline and to harbors. CN's are an embodiment of this kind of knowledge. In this report, CN's are used in the domain of image understanding to illustrate the more general principles involved in continual search-space refinement.

A CN is composed of nodes representing objects or object locations and arcs specifying operations which express constraints between them. Each constraint serves to determine the object location more precisely within the image by limiting the possible area where the feature could occur.

Specifically, a CN is the data structure which we use to represent these constraints. Normally, a CN serves as a data source giving the feature's location. However, since the facts which limit the possible location of the object are explicitly encoded by the nodes of the CN, if the location is not known when the CN is interrogated, then an evaluator can use the CN to compute the feature's location. When a CN is evaluated, the evaluator uses the knowledge encoded in the structure of the CN and data available to compute the most likely area in the image where a particular feature may be found. So, Constraint Networks offer an inexpensive way to eliminate large parts of the image from analysis by explicitly indicating where to look next when given some contextual clues. In many scenes, information about the location of one feature can specify the locations of others in the picture [Garvey].

However, when modeling real world constraints, we need not limit the kind of knowledge used to simple relations of feature locations within an image. CN's can also utilize additional information about the domain of interest. We might also know, for example, that docks have a normalized albedo above some determined value for aerial photographs. Knowing this fact would immediately reduce our searching for docks to those areas of the scene which have a reflectivity greater than the value specified. Domain-specific knowledge of this sort can also be represented as a constraint which limits the range of values assumed by a feature description.

While these kinds of knowledge can also be represented as a group of assertions, each encoding a single constraint, we choose the network format for representing our constraints because 1) CN's are a formal structure which can explicitly encode the relationships between features easily, 2) it provides a facility for optimization of evaluation and sharing of partial results, and 3) it is a simple way to compose complex constraints from primitive constraints in a straightforward manner.

III. Constraint Networks: Structure and Function

As we have established, a CN describes constraints which are known about the location of objects in a particular image. This knowledge is embedded in the form of a connected network of nodes. Each node represents some feature or object in the image. The network is composed of nodes of three types:

Feature nodes - are the handle by which the Constraint Network is accessed by some larger image understanding system. The CN under a feature node embodies the knowledge which describes a particular feature in an image. A Feature Node has attached to it (as sons) CN's which are alternative encodings of the possible locations of the feature in the image. These CN's may be thought of as different strategies for finding the feature. Yet, a CN is not a completely procedural mechanism for representing knowledge, for associated with each node in a CN is the result of the evaluation of the CN below that node. This also holds true with the feature node; if the entire CN has been evaluated below the feature node, then the feature node contains the result of that evaluation. In this case, "evaluation" of the CN becomes a simple lookup. In other words, a CN is a "compute when required" structure which minimizes the amount of processing that it must perform. This is similar to the idea of "memo functions" as suggested by [Michie].

Operation nodes - are the nodes encoding the various constraints which are placed on the feature being searched for in the image. An operation node gets input from all of its sons and then applies the operation it represents on that data, thus realizing the constraint. Operation nodes represent the geometric relationships between features and are operations chosen from some system-defined set of primitives.

Data nodes - are the terminal nodes of the network. That is, they have no sons and always evaluate to data. Data nodes supply an unevaluated network with initial image data to operate on; they usually correspond to locations or image features which are relatively easy to determine.

The nodes of a CN can all potentially hold data. This capability is used to store the partial results found during an evaluation of the CN. As a result, all nodes are always in one of four states. A node is UP-TO-DATE if the data attached to it is a valid instance of the feature in the image. A node is OUT-OF-DATE if no data is attached to the node (i.e. it is not known if this primitive feature exists in the image); the node can be NONE-THERE if it is known that no primitive feature of this type exists in the image, or finally the node can contain information which is HYPOTHESIZED (the result of the evaluation of a CN and may not truly exist in the image). Each different status affects the results of node evaluation, and the way that results are handled by any nodes which use the result. A node that is OUT-OF-DATE returns a value which indicates that the answer may be anywhere in the UNIVERSE of the image. An UP-TO-DATE node explicitly points to the feature in the image. A node which is HYPOTHESIZED determines a location in the image, but the data may or may not locate the specified feature since it is the result of the evaluation of another CN. A HYPOTHESIZED result is considered the most likely location of a feature until the validity of the HYPOTHESIZED data can be verified. Finally, a node which is in a NONE-THERE state indicates that the feature simply doesn't exist in the image, or that all instances of that feature in the image have already been bound to other nodes which describe this feature. (This distinction is easy to make, but is only performed if required.)

IV. Constraint Types

Constraints in the CN world are expressed by geometric operations on data. The operations encoding primitive geometric constraints are chosen from a set of basic operations which describe transformations on areas, describe relationships between areas, specify shapes and the like. The function of the operations in the primitive set is to provide the CN builder with enough tools to describe flexibly and naturally image areas and their relationships with other image areas. Although the number of potential operations is quite large, we have found that a small number of primitives (about twenty) suffice for most of our descriptive tasks.

In our system, the primitive set is made up of four different types of operations.

Directional operations specify where to focus attention. Operations such as LEFT, REFLECT, NORTH, UP and DOWN all constrain the sub-image to be in a particular orientation to another feature.

Area descriptions specify a particular area in the scene that restricts a feature location. For example, CLOSE-TO, IN-QUADRILATERAL, and IN-CIRCLE define areas at some location in an image of interest.

Set operations permit areas to be handled as point sets of pixels. These operations, such as UNION, DIFFERENCE and INTERSECTION make very complex image areas far easier to describe.

Predicates on areas allow features to be filtered out of consideration by measuring some characteristic of the data. For example, a predicate testing WIDTH, LENGTH or AREA against some value would restrict the size of features in consideration to be only those within a permissible range.

In actually constructing a CN, the builder is not limited to building CN's from purely primitive operations. Since a CN represents an implicit description of an area in the image, it can be used by other CN's as an operation for locating a feature. This allows the CN builder the ability to form very complex CN's by building upon previous work.

V. Describing an Object Location by Constraints

By using a combination of geometric constraints we can describe the expected location of most features that we would like to find in a scene. We can use the location of the aeration tank in a sewage treatment plant for an example.

The aeration tank is a rectangular tank surrounded on either end by circular sludge and sedimentation tanks.

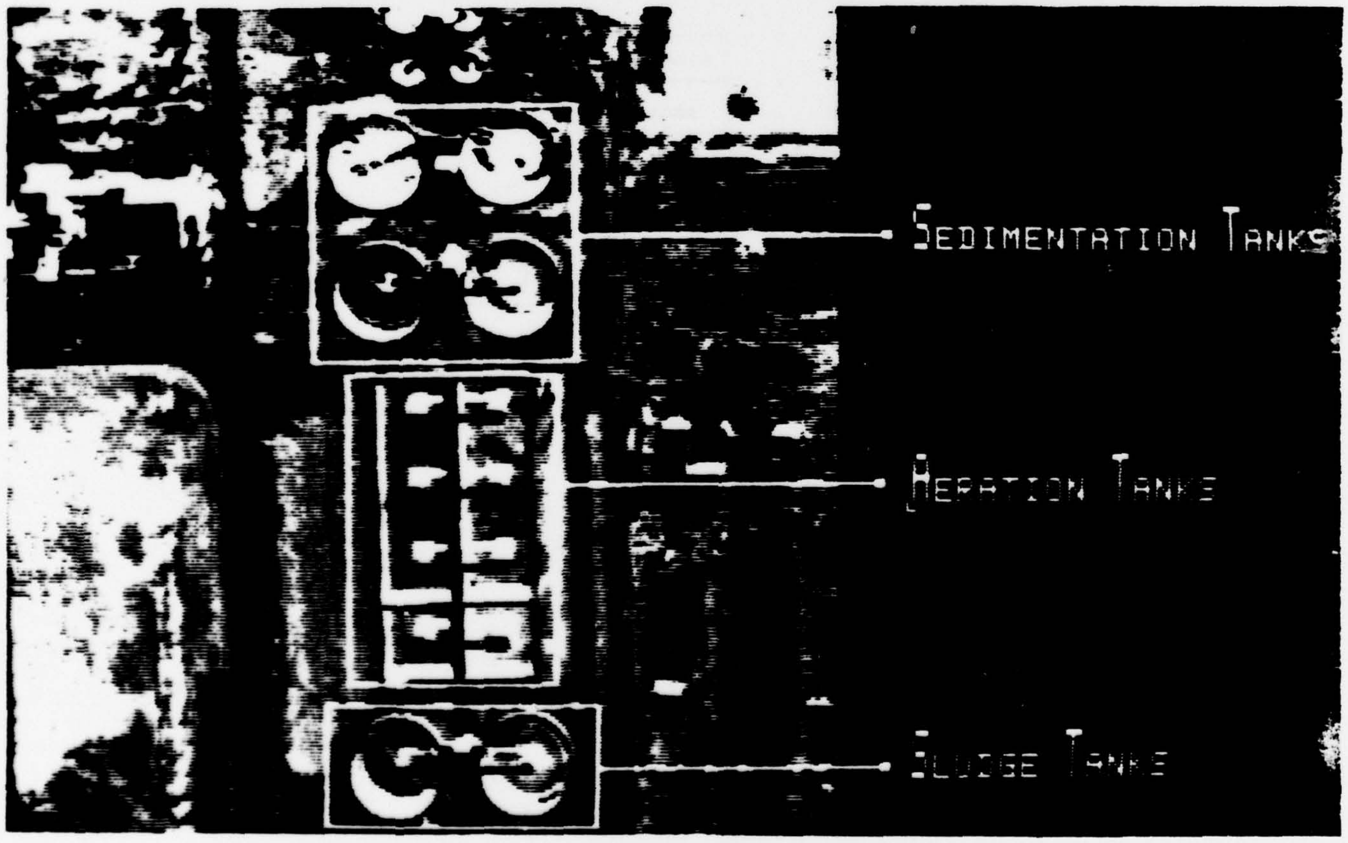


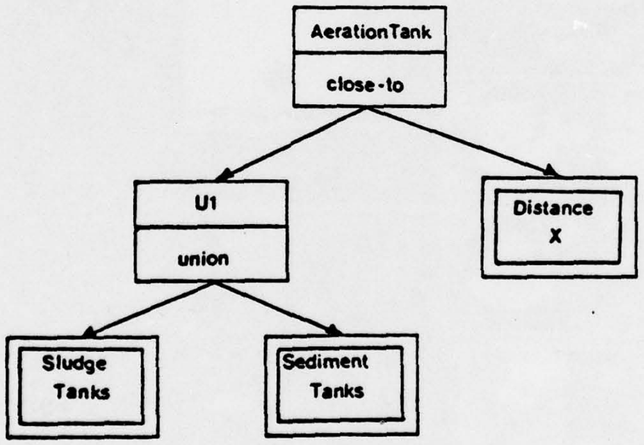
Figure 1 - Aerial view of a sewage plant

As a general rule, sewage flows from the sedimentation tanks to aeration tanks and finally through to the sludge tanks. This design permits us to identify and use the following types of constraints on the location of the aeration tanks.

<u>Constraint</u> <u>1:</u> "Aeration tanks are located somewhere close to both the sludge tanks and the sedimentation tanks."
<u>Constraint</u> <u>2:</u> "Aeration tanks must not be too close to either the sludge or sedimentation tanks."

These constraints seem reasonable because we understand that in a design of a sewage plant, the various components of the treatment process will be close together for practical reasons. And, since the aeration phase occurs between sludge treatment and sedimentation collection, we can expect to find the aeration tanks between the sludge and sedimentation tanks.

The first constraint explicitly relates the possible location of the aeration tank with the location of the sludge and sedimentation tanks. This relationship would be encoded in a CN as:



<u>Figure 2 - CN encoding of Constraint 1</u>
(see Appendix 3 for notation)

When this CN is evaluated, the top node would have data expressing that part of the image which satisfies Constraint 1; that is, that portion of the image which is within some distance X of both the sludge tanks and the sedimentation tanks. (X could be found empirically. It could also be the result of another CN which might consider the tank's diameter or the camera angle of the image.)

Similarly, Constraint 2 would be encoded as shown in Figure 2.1 below.

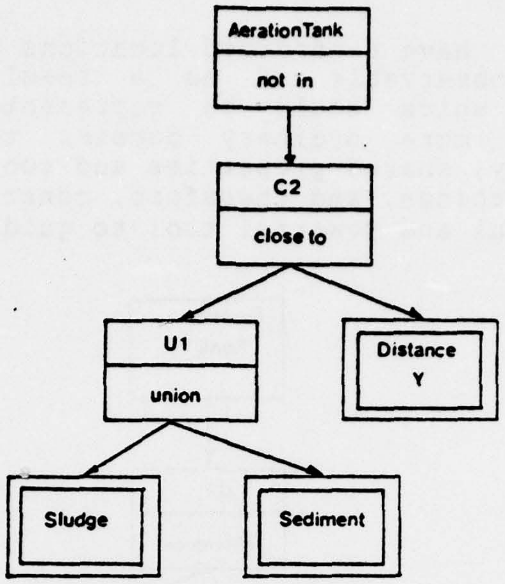


Figure 2.1 - CN for Constraint 2

The entire network describing the probable location of the aeration tanks must embody both of these constraints. Constraint 1 determines an area which is close to both groupings of tanks and Constraint 2 says that we should disregard a portion of that area. If we think of the image as a point set (that is, a list of pixels in some order which makes up the image), the function to perform would be a differencing operation to remove the area given by the second constraint from that specified by Constraint 1. Figure 2.2 shows the final CN which incorporates both constraints.

Of course, there could be places where the aeration tanks might be located very far away or perhaps violate some other constraint. It is important to note that the constraints are only the most likely limitations on an object. They can work very well for stereotyped scenes, and might fail to perform in novel situations. The cause or ontology of the constraint is unimportant here. (See Section IX on Future Work.) Simply that such constraints exist and can be utilized by an image understanding system is important in this context. We believe that geometric constraints are adequate for the kind of features we may wish to locate in an image because things of complexity usually exhibit differentiable parts related and interconnected by their functions or by their existence in a common milieu. Because of this, these relationships often can be expressed in terms of their form, or, put another way, by geometric descriptions which relate parts of the whole. Domains which exhibit weak relationships between discernable features would cause Constraint Networks to perform poorly. This can occur, for example, when attempting to analyze microphotographs of a slide smear.

The objects have randomized locations on the slide to make them easily observable and as a result, demonstrate few connections which could be represented as constraints. However, in more ordinary domains the connections of functionality, shared properties and cooperation affect most commonplace things, and therefore, constraining attention is often a useful and powerful tool to guide search.

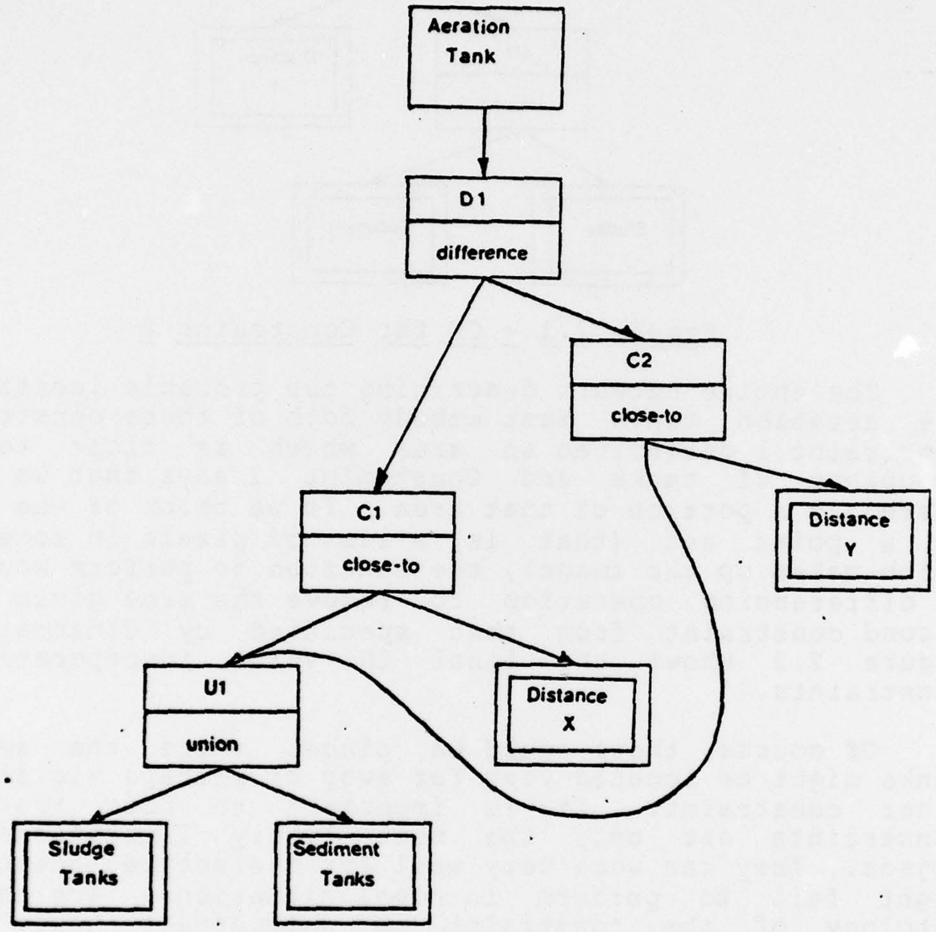


Figure 2.2 - Complete CN for aeration tank location

VI. Evaluation of Constraint Networks

A Constraint Network is evaluated by a special purpose evaluator working top-down in a recursive fashion, storing the partial results of each constraint at the topmost node associated with that constraint, with a few exceptions.

In Figure 3 we give an example of a Constraint Network which is to find the most likely site for new construction on a new oil tank farm. We know from experience that the construction of new oil tanks is generally going to occur near other, older oil tanks. But since we only wish to find new construction, we can eliminate the old oil tanks and their immediate vicinity. With this in mind, we can interpret the CN in Figure 3 to represent

"New construction around oil tanks is usually found near the oil tanks, but not on the oil tanks themselves."

The computation required is not difficult to see. We want to compute the union of the oil tanks, and subtract that area (the oil tanks themselves) from the union of the areas near the oil tanks. This should reduce the image area being searched from the entire image to a considerably smaller "swiss cheese" area in and around the tanks. (As a further refinement to the CN we could also remove permanent structures which are within this region. The CN has been simplified for discussion's sake.)

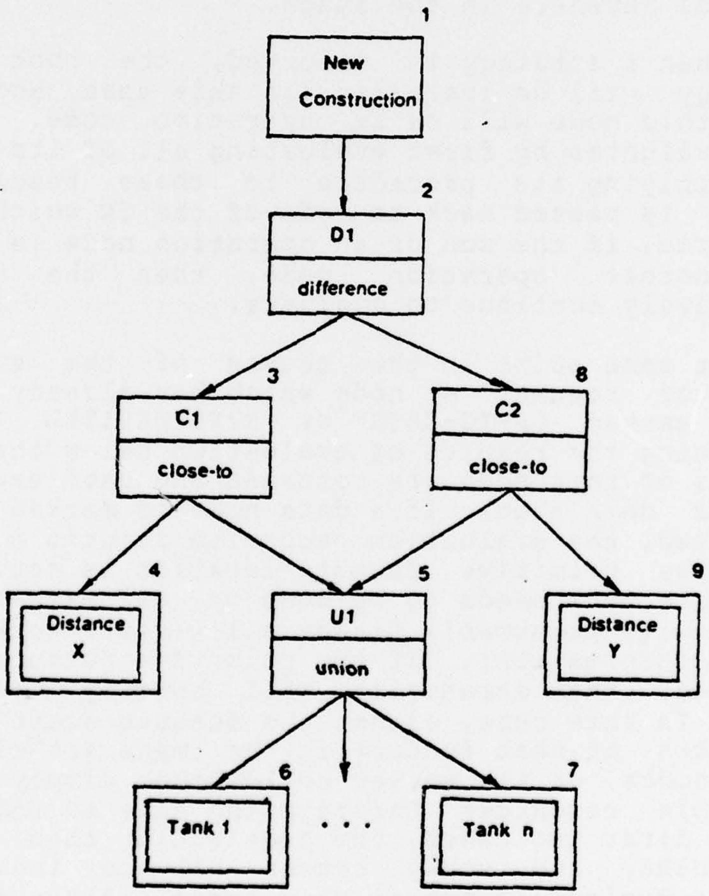


Figure 3 - a CN for new construction site in an oil field

We can think of a CN's evaluation in the following way. Feature nodes may have several sons, or sub-CN's, each encoding a separate strategy. A particular strategy is selected by a strategist as described in [Lantz, et al]. The strategist computes the utility of each strategy attached to the feature node and based on this estimate the most desirable strategy is selected. The strategist's measurement of utility is based on both an *a priori* measurement of the algorithm's effectiveness and an assessment of the status of the data present in the body of the CN.

On this basis, the strategist interested in the feature node chooses a strategy and begins to evaluate the CN. In Figure 3, the Feature node (Node 1) has only the single strategy mentioned above. The strategist would be forced to select this strategy for evaluation. In the more general case, strategies of a feature node are evaluated until an answer is obtained or all strategies are exhausted. If all strategies are evaluated, and none is able to return a result, then the entire image is returned. The feature could be anywhere in the image.

When a strategy is selected, the root node of the strategy will be evaluated (In this case, Node 2). In most CN's, this node will be an operation node. An operation node evaluates by first evaluating all of its arguments, and then applying its procedure to those results. Its own result is passed back to node of the CN which evaluated it. Of course, if the son of an operation node is a feature node or another operation node, then the evaluator will recursively continue to evaluate.

At some point in the course of the evaluation, the evaluator reaches a node which has already been evaluated and is marked UP-TO-DATE or HYPOTHESIZED (and therefore containing the results of evaluation below that point). The results of this node are returned and used exactly as if it were a data node. If a data node is marked OUT-OF-DATE is evaluated, the evaluation mechanism returns a result stating that the primitive feature location is not specified, and that more work needs to be done by an executive procedure (which will presumably direct a low-level worker to find the needed information). If the primitive feature is then not supplied, the strategist will specify the status of the node. In this case, either the feature doesn't exist, all instances of that feature in the image are already bound to other nodes, or the worker could have simply used up all available resources before being able to return an answer. In the first two cases, the node would then be marked as NONE-THERE, and would return NIL to indicate that the feature desired is not in this image. Alternatively, if the worker has exhausted its resources, but has not yet determined the status of the feature described by the node, the node will remain OUT-OF-DATE and have the entire image as its value. This indicates that the feature could be located anywhere in the image. At a later date, processing can resume from this node without having to recompute the part of the tree which was already processed. Finally, a node marked HYPOTHESIZED has data which was inferred by a CN somewhere down the line of inference. HYPOTHESIZED data can and is used to make inferences, but the results of all inferences based on hypothesized data are marked HYPOTHESIZED as well.

VII. An Example of CN Evaluation

Figure 3 is the graphic representation of the Constraint Network which computes the area of probable new construction of tanks at a tank farm site. This CN embodies the strategy (given above) of "New construction of oil tanks is found near the old oil tanks, but not very close and not on the oil tanks themselves".

To evaluate the CN, the evaluator would begin with Node 1, the feature node, and discover that it was initially OUT-OF-DATE. The evaluation routine would then evaluate the strategies attached to the node in the order specified by the strategist. In this case we have a simple choice - only a single strategy exists. This process would then continue until a strategy returned a valid response, or until the strategist decided that continued evaluation would become too expensive to pursue any further. The evaluator would proceed to evaluate the strategy embodied in the remainder of the CN. To do so, it would traverse the graph, returning a value only when it finds an UP-TO-DATE or NONE-FOUND node, or if it simply runs out of alternatives (then returning NONE-FOUND too).

To evaluate the strategy, the evaluator would then evaluate Node 2. Since it is an operation node, each of its sons must be evaluated before it can apply its constraint. Now the evaluator has a choice between Nodes 3 and 8. Since each node is realizing a single constraint, the order of evaluation is unimportant. We choose preorder for the sake of discussion. Continuing on in this fashion, the evaluation reaches Node 4. This node gives a distance measure to Node 3. It quantifies the meaning of "near the oil tanks". Again, this value can be found empirically, or it too can be the result of another CN. In this CN however, the distance is given by a data node which is marked UP-TO-DATE. Upon evaluation, this node returns an argument to Node 3.

This procedure of evaluating sons and returning continues until the evaluation reaches Node 8. When Node 8 evaluates its sons, it first attempts to evaluate Node 5. However, Node 5 has already been evaluated from Node 3 and marked UP-TO-DATE. Since the node has already been visited, the evaluation of Node 5 returns its result to Node 8 without having to re-compute.

Results from each evaluation are propagated up the network and each node's result is stored at that node and the node is marked UP-TO-DATE. Finally, the result of Node 2 is available and Node 1 can be updated with its correct result. The evaluator then returns with the Feature node's result.

Since we have not yet verified the presence of new construction in the area returned by the CN, and are therefore uncertain of its exact status, we attach the label HYPOTHESIZED to the data now stored at the Feature node, Node 1.

VIII. Attachment of Data to Data Nodes

As we shall see below, every Constraint Network must begin with some data. However, when we write the CN's, we don't know how much or what kind of data will be given at evaluation time. How does the CN find its initial data set?

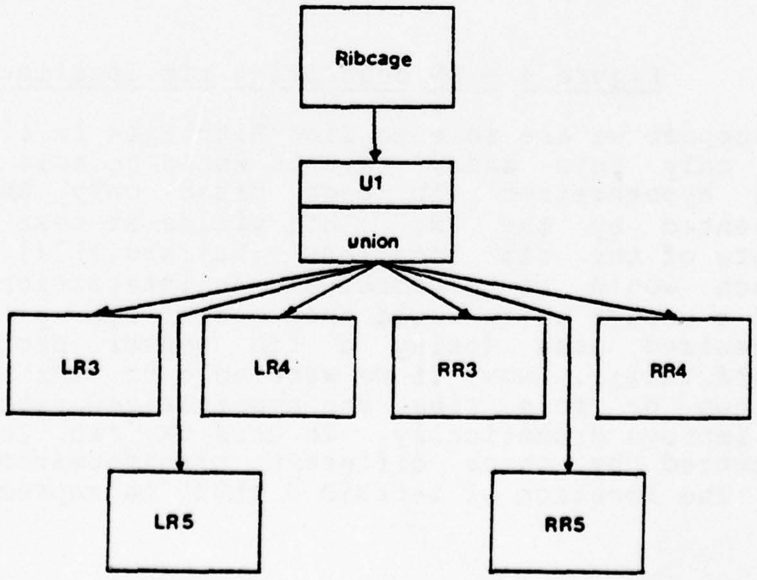
Since it represents generic knowledge, the CN is constructed separately from the image, and, for a useful system, the CN must be able to work over a wide range of inputs from many sources. To solve the problem of identifying nodes in the CN with instantiated features in the scene, we have developed two methods of run-time attachment of data. Each node in the CN, whether it is a Data, Operation or Feature node, has a descriptive name associated with it. At run time, when data is added to the system, it is bound to a node based on this description. This object description is CN dependent and is simply a text description of the item as it appears in the scene. It is chosen from a list of such descriptors which apply to a particular CN. For example, "oil tank 23" would be an object peculiar to one image and could be bound to the generic node "oil tank" in the CN. Data can also be bound to specific nodes in a CN by structurally matching parts of the CN to a description of the area expressed in the relational primitives. That is, the data being input is described by a sub-CN specified by the agent delivering the data. This sub-CN is then matched to portions of the the existing CN by graph-matching techniques and the data is bound between corresponding nodes. In both cases, the binding is done only on demand. Demand occurs whenever an OUT-OF-DATE node is evaluated. When this happens, the data which has been added is examined to see if any item matches either the name or the structure of some node in the CN. If a free datum does, its value is attached to the node which is being evaluated; and the datum is removed from the new-data pool.

Primitive feature data could come from a map, as in [Barrow], from a special purpose "worker" routine, designed to find the primitive features that are easy and quick to find; (as in Ballard's Right Rib 4 finder [Ballard,TR11]), or from some other external source. This gives the CN a basis upon which to direct search for the features in the image.

In keeping with the generic nature of the uninstantiated CN, no data is attached to any node. Whenever a new feature is bound to a node, those results which depend on it as a part of their strategy should be re-computed. To separate out only those parts of the CN which are dependent on the newly updated node, all of the nodes above an updated node are marked OUT-OF-DATE. The effect of this is to invalidate all of those HYPOTHESIZED features whose computations were based on what has now become incorrect data. Since the evaluation process proceeds only past nodes which are marked OUT-OF-DATE, updating the status of only the invalid nodes minimizes the amount of recomputation which will have to be done to re-evaluate the CN, that portion of the CN which must be recomputed from that which is still valid.

IX. Use of Partial Knowledge

Usually, each CN starts with some data in the form of features already located in the image, numeric values specified, etc. However, the number of UP-TO-DATE nodes may vary from as few as one, to as many as every node in the network. One of the key features of CN's is that their performance improves as the number of UP-TO-DATE nodes increases. The accuracy to which a CN can compute an area in the image corresponding to the desired feature depends to a large part on the number of UP-TO-DATE nodes which it uses during its evaluation. Figure 4 is a CN describing the expected locations of ribs in a medical image based on the locations of the neighboring ribs either across the chest, or up and down the rib cage. This is an unusual CN: it has no explicit data nodes which represent ribs, but since each node can contain data, we can bind data to an operation node or feature node exactly as we would a data node.



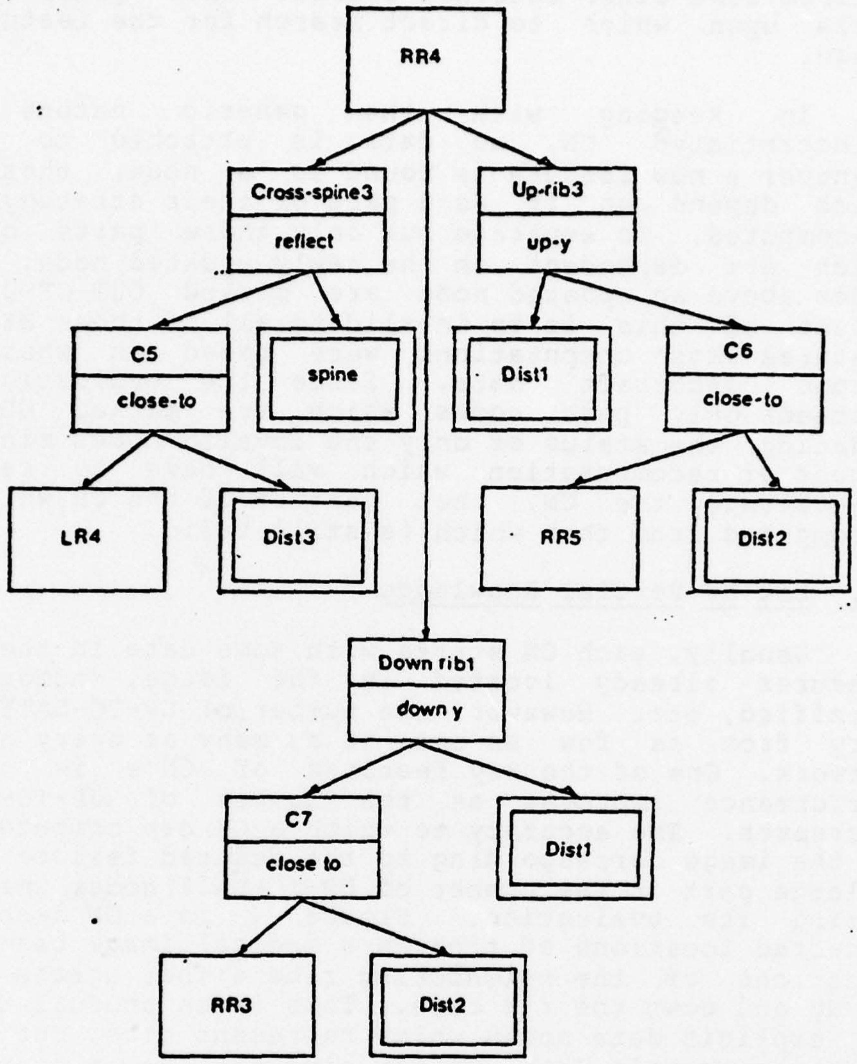


Figure 4 - CN describing rib locations

Suppose we are able to find RightRib4 in a chest X-ray. Using only this data, the CN would be able to return an entire hypothesized rib cage based only on the model represented by the CN. This yields at best only a crude estimate of the rib locations [Ballard,TR11]. A better approach would be to coordinate an interaction between the CN and a worker which would then look for a rib in the hypothesized area (using a rib worker procedure a' la [Ballard,TR11]). Now, if we were able to initiate the CN with two or more ribs, the hypothesized ribcage location would improve dramatically. In this CN, rib locations are represented by three different transformations of known ribs. The location of LeftRib 3 (LR3) is represented to be

a) RightRib 3 reflected about the spine, b) LR2 shifted down the rib cage by one inter-rib distance, or c) LR4 shifted up by a similar amount. Each of the three strategies has a different utility. The reflection of a rib around the spine has a good chance of constraining the rib location very tightly. This is not as true for the down-rib operation, and the worst case occurs when an upward rib is hypothesized from the source rib. [Ballard,TR11]

Each constraint operation returns an area of likelihood based on whatever results it has available for use. If this data were the result of a previous conjecture, and that the result of a conjecture before it, uncertainty about the exact location of the feature location accumulates. As an empirical result, however, we have found that this accumulation of uncertainty is not a very serious problem for domains where the chain of inferences is short, or the relative positioning of features is restricted. In the ribs world above, for example, the successive rib locations are highly constrained with respect to the ribs around them and therefore the uncertainty in hypothesized rib locations never becomes very large, even after passing through a long chain of reasoning. One might find examples where the accumulation did become so great as to render the result of many successive constraint operations of little value. However, this problem can be obviated by appealing to the higher level strategist to verify the sub-CN's hypothesized results and thus make them into reliable information sources before building upon their results [Lantz, et al].

Since partial results are maintained in context, the utilization of new and corrected data becomes a simpler task. With each operation explicitly denoted and its results available, re-evaluating a given operation reduces to finding the node which made the last correct computation and then attempting either an alternate strategy or supplying new information to be evaluated, in a new and more correct fashion. Also, since the addition of new information to the CN marks OUT-OF-DATE only those computations which are dependent on that information, only they will need to be re-evaluated.

X. Grain Size of Constraint Networks

In the hierarchy of data structures produced during image analysis, the level of effective operation of Constraint Networks seems largely limited by the nature of the expectations incorporated into the CN. The constraints used are static by nature and their applicability seems limited to high-level concepts. We have found that CN's tend to become difficult to manage effectively at low levels of domain representation and inferencing. In the vision domains we have studied, low level processing such as region growing, edge following or the like do not seem easily amenable to representation as Constraint Networks. CN's can easily provide an adequate mechanism for line following when the edges are of high contrast. But in noisy environments and at small grain size, the strong interconnections between features rapidly become weak, reducing the basis on which Constraint Networks operate.

XI. Future Work

A desirable future extension of Constraint Networks would be to incorporate some notion of the connection between structure and function in computing an object's most likely location in a scene. This would initially require that the CN perform inferencing of a different sort about the structure of a feature. Currently, the knowledge in a CN is structurally oriented. It describes the location of an object based solely on its relationships to other objects in 3-space. Comprehension of the functional connections between objects would greatly increase the robustness of feature location.

The dynamic data attachment mechanism is fairly expensive if the semantic description part fails, since it involves sophisticated graph matching between the input description and portions of the CN. This could be a substantial area for improvement.

Constraint Networks can also be used as a knowledge source describing the relationships between objects in an image. In this use of CN's, they act as a static representation of the interconnections between items, separating features from their functions in CN's. We have made some preliminary efforts in this direction, attempting to categorize the nature and manner in which non-geometric inferences could be made from the structure and contents of the CN.

XII. Summary

In this paper I have attempted to illustrate how Constraint Networks operate, what kinds of things they can be used for and to indicate their adequacy as goal-oriented directors of a search process. Constraint Networks have been shown to be useful in directing specialist routines in a model-driven search for features in both medical and aerial image understanding.

In addition, I have shown a few of the properties characterizing Constraint Networks, including: improving accuracy with additional data, automatic attachment of data to procedures by semantic description, maintenance of partial results, sharing of partial results by matching on their structural description, and "compute-when-required" behavior minimizing the total processing performed to extract a feature location.

Appendix 1
Operation of a CN Evaluator Program Example 1

Below we show the results of processing by the Constraint Network given in Figure 5. This trace illustrates the capability of CN's to improve the hypothesized rib locations with differing amounts of initial information.

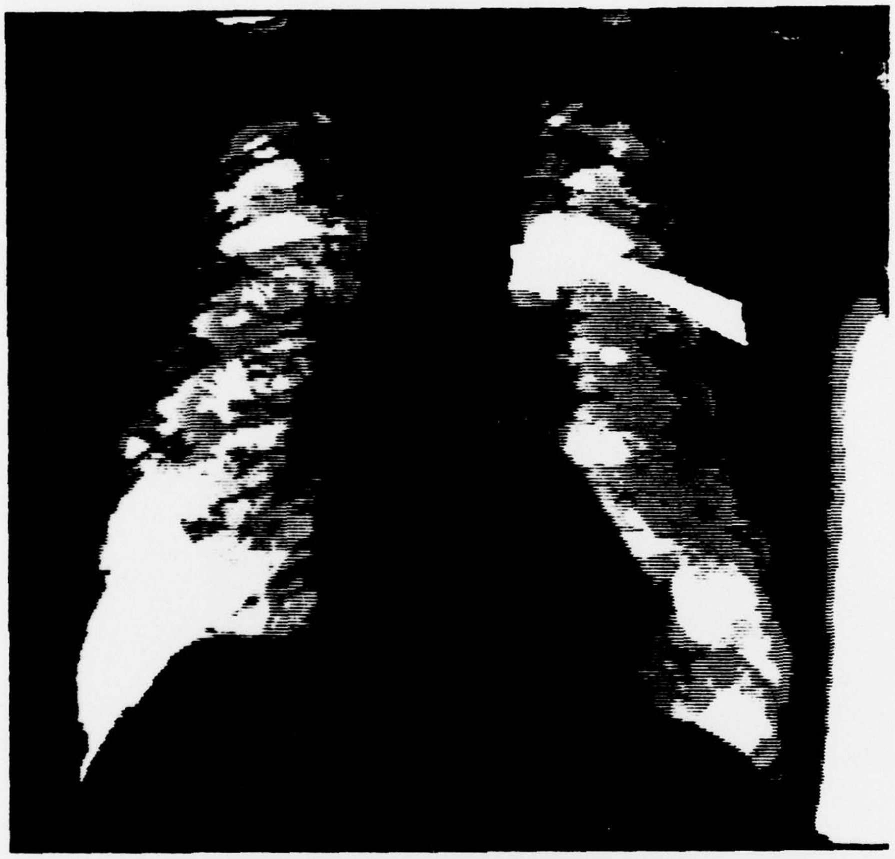


Figure 5

Figure 5 shows the initial data given to our simple model of a rib cage. In this case, we have specified only the location of right rib 4 (RR4). This data could have come from a human source or specialist workers designed to find ribs with a high *a priori* reliability.

If we now evaluate the CN, the remaining five ribs will be hypothesized using three different strategies. Figure 6 shows this result. The choice of strategy is dependent on its expected utility and is made by a strategist overseeing the evaluation.

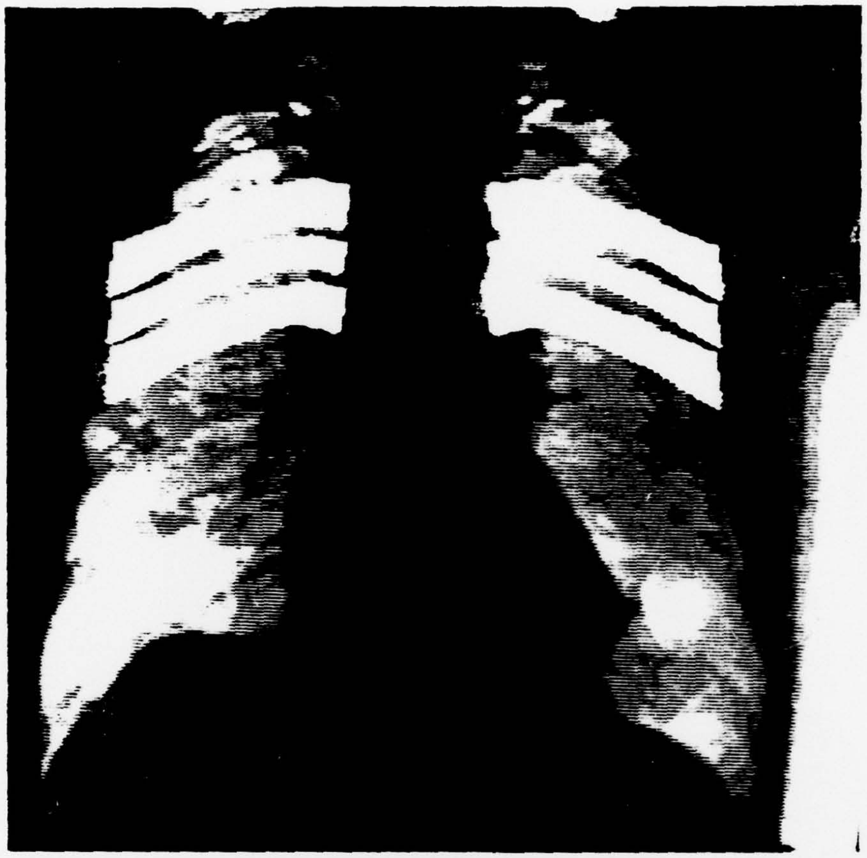


Figure 6

LR4 is hypothesized with the "good" cross spine reflector model, which statistical studies have shown will give us good matches for the ribs [Ballard]. LR5 is computed by using the down translation from the location of LR4. While we know this prediction will give us a reasonable estimate of the rib locations, we can see that it is not nearly as precise as that given by the reflection. The remaining ribs are hypothesized from this data by continuing the up and down translations from the hypothesized results of other CN's.

In Figure 7 the CN is initialized with a larger data set, this time it is given LR3, RR4 and RR5. Starting the network with this much data is not entirely implausible. We might, for example, have computed these rib locations in a previous session or they might be the result of a different part of the image understanding system which was able to compute these rib locations easily and could share the results with the CN model.

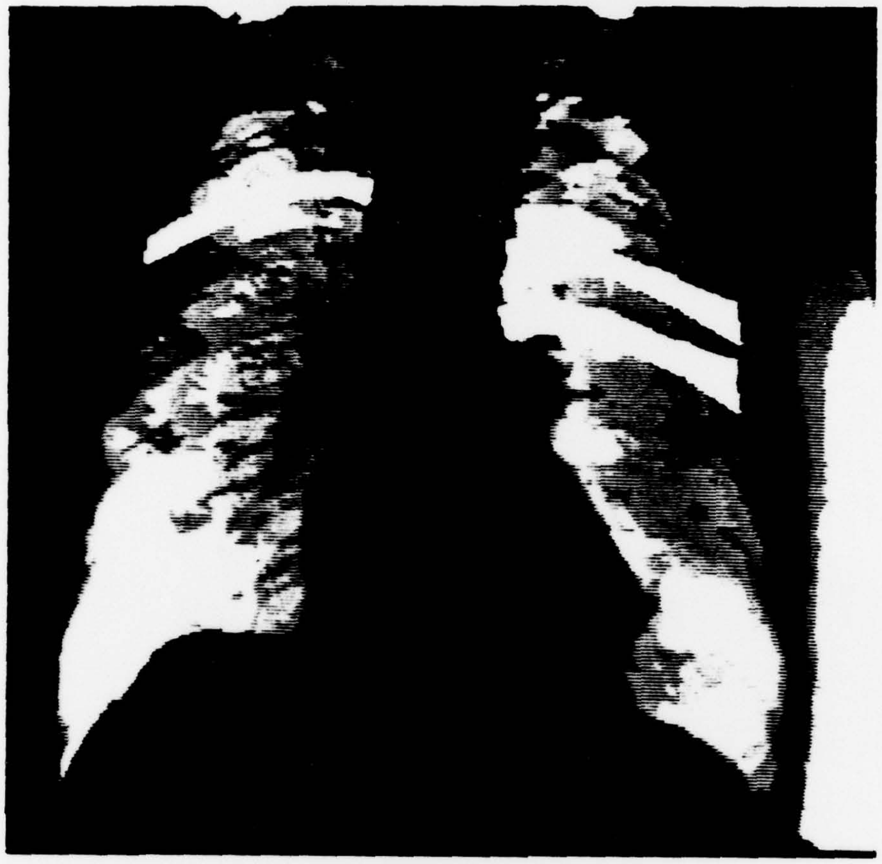


Figure 7

If the CN is now evaluated it would compute the result shown in Figure 8.

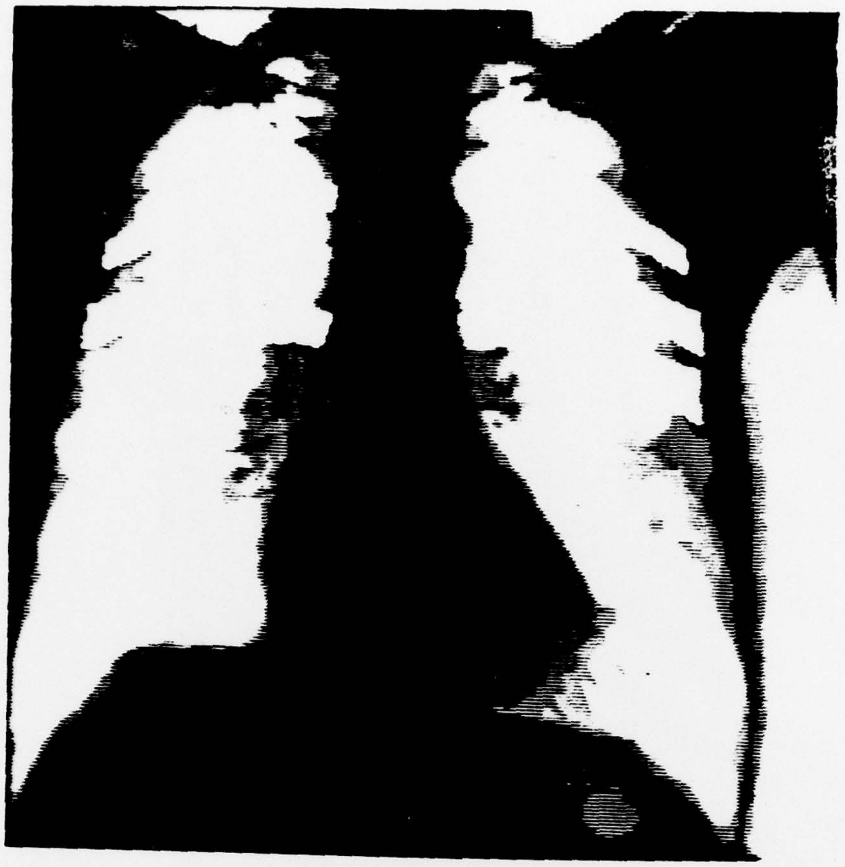


Figure 8

If we compare Figures 6 and 8 we can see the improved performance with additional data. In particular, RR3 and LR5 are confined to a smaller area in Figure 8 since the CN could use the reflection model for rib location rather than the less effective up or down translations. Improvement in the quality of the hypothesized regions comes about as a result of being able to use the superior strategies for predicting rib locations, and not having to rely on a a long chain of hypothesized results based on the poorer quality strategies.

Example 2

Another way in which Constraint Networks improve with increased amounts of information is by being able to remove areas from consideration based upon the added data.

Figure 9 shows the initial data given to the aeration tank CN from Figure 2.2. In this case we are able to start the CN with only a single sludge tank and a single sedimentation tank.

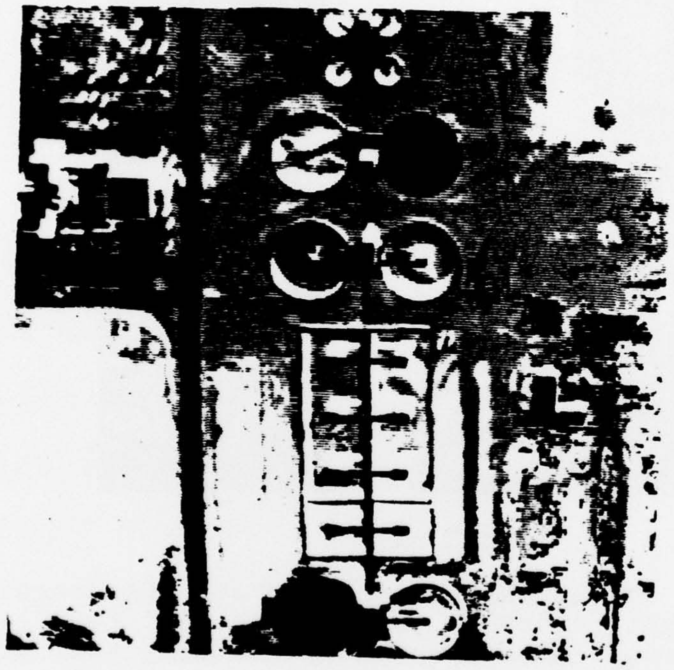


Figure 9

When the CN is evaluated, the result is that shown below in Figure 10.

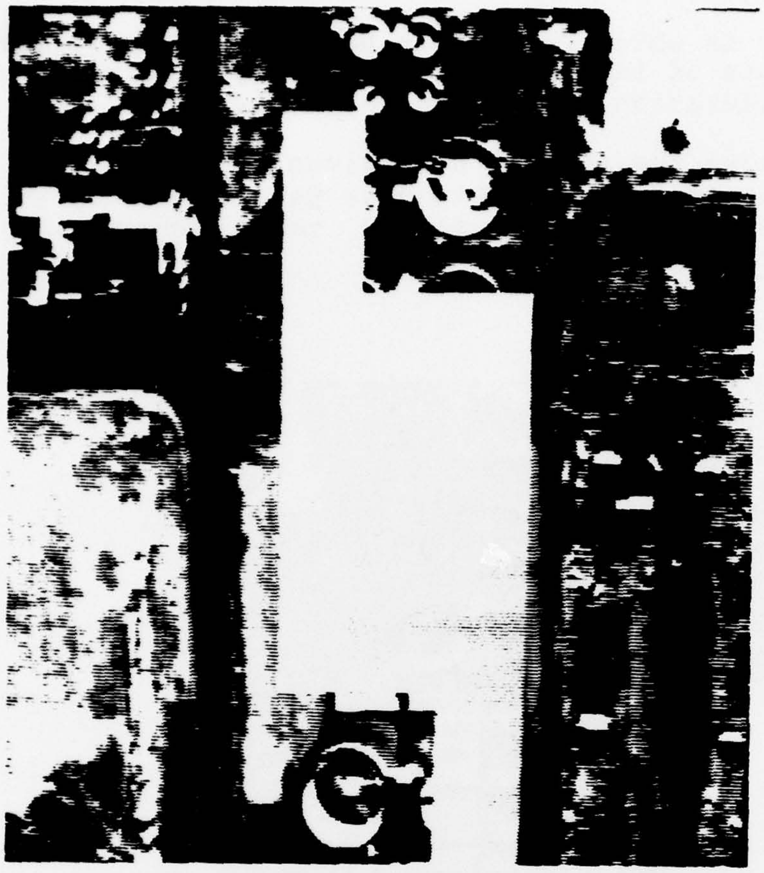
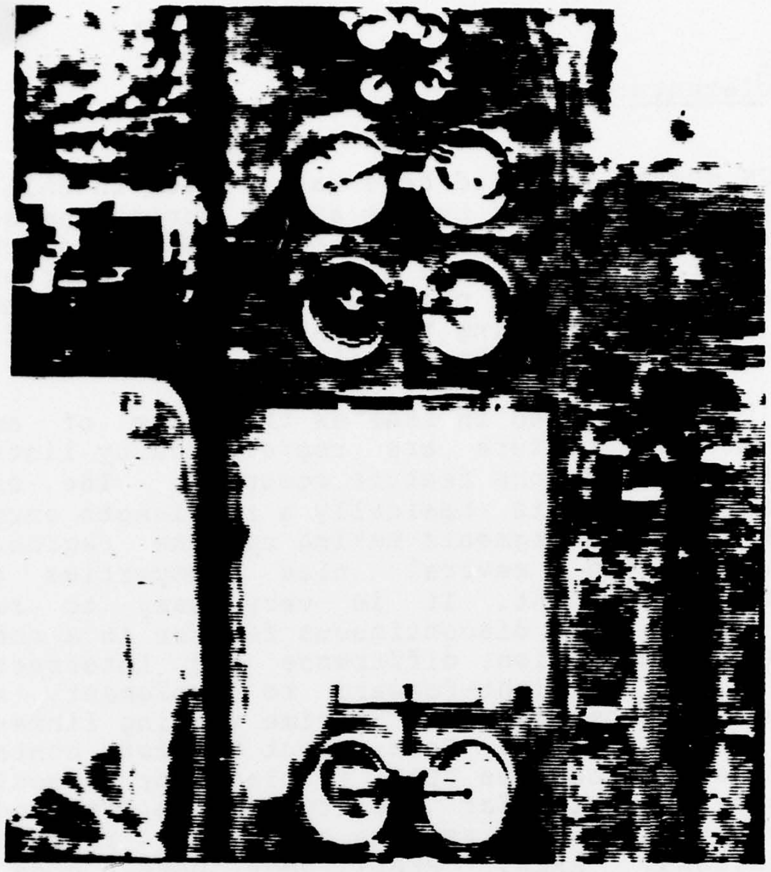


Figure 10

If we now add to the CN the location of the remaining sludge and sediment tanks in the picture, and re-evaluate the network, the result more accurately reflects the actual location of the aeration tanks.



<u>Figure 11</u>

Appendix 2
System Implementation

The CN system reported here was written in SAIL at the University of Rochester by the author during the summer of 1978. It consists of three programs - CNGEN, the Constraint Network Generator; PIC, the data set constructor and EVAL, the CN evaluator and test program. The programs communicate via disk files containing the LEAP world which defines not only the CN's, but the data as well.

Data is represented in LEAP as the datum of an item. Features in the picture are represented by lists of the pixel locations which the feature occupies. The canonical representation used is basically a run-length encoding of horizontal scan-line segments making up the region. This representation has several nice properties from an implementation viewpoint. It is very easy to represent multiple areas, or a discontinuous feature in a scene in a single list datum. Union, difference and intersection of areas are all straight-forward to implement, and the merge-like algorithms used run in time varying linearly with the size of the regions. Facts about the data contained in a data node are encoded as LEAP triples (or associations) which state a particular quality of the data node. The triples assert facts such as data type (the representation used; INTEGER, AREA, REAL) node name, node status (HYPOTHESIZED, OUT-OF-DATE, NONE-FOUND, UP-TO-DATE), and which nodes are sons or fathers of a given node.

Constraint Networks are also represented in LEAP. In the same way, LEAP triples are used to represent the Father-Son relationships between nodes in a network and to associate the various node states with each node. In LEAP notation, a node which was out of date would be -

VALIDITY of NODE is OUT!OF!DATE

The process of generating the CN's and saving their structure onto disk is done by the CNGEN program. This program runs interactively on a Grinnell color display, allowing the CN builder to see the CN's as they are being made. The program permits the builder to edit, create and delete CN's easily and quickly. The desirability of such a facility for semantic networks was recognized in [Brachman].

PIC takes digitized images and creates the initial data nodes for the CN's to evaluate. PIC can create arbitrary shapes interactively by using various sizes of circles, arbitrary quadrilaterals and lines. Complex shapes are formed by merging together smaller pieces of the shape to form the final region.

Finally, EVAL performs the evaluation of the CN's. EVAL accepts data sets and CN's on demand. It offers tracing facilities which display the result of the evaluation of each node in a different color. This facility makes it easy to follow the inferencing patterns of the CN in use and permits an easy way to follow the actions of the strategist.

Appendix 3 Constraint Network Notation

On the Grinnell color display, a CN is a multi-hued splendid thing. Feature nodes are red, Operation nodes blue and Data is green. As an alternate representation for a black and white world, the nodes are labeled as below -

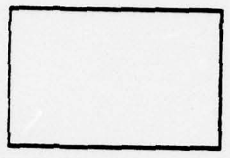


Figure 12 - Feature Node

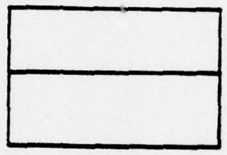


Figure 12 - Operation Node

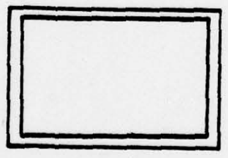


Figure 12 - Data Node

Acknowledgments

This research was partially supported under DARPA grant N00014-78-C-0164. My gratitude goes to Christopher M. Brown for invaluable editorial assistance.

References

[Ballard,TR11] Ballard, D.H. "Model-Directed Detection of Ribs in Chest Radiographs", TR11, Computer Science Department, University of Rochester, March 1978

[Ballard, et al] Ballard, D.H., C.M. Brown, and J.A. Feldman. "An Approach to Knowledge-Directed Image Analysis", TR21, Computer Science Department, University of Rochester, September 1977; also in Proc. 5th IJCAI, MIT, August 1977

[Ballard] Ballard, D.H. personal communication on studies in rib-detection algorithms

[Barrow,et al] Barrow, H.B., Amber, A. P., Burstall, R. "Some Techniques for Recognizing Structure in Pictures", Frontiers of Pattern Recognition (ed. Watanabe, S.), pg. 1-19, Academic Press, New York

[Brachman] Brachman, R. J. "A Structural Paradigm for Representing Knowledge", Ph.D. Thesis, Harvard University, 1978; also as BBN Report 3605, May 1978

[Garvey] Garvey, T. D. "Perceptual Strategies for Purposive Vision", SRI Artificial Intelligence Center Technical Note 117, September 1976

[Lantz,et al] Lantz, K. A., C.M. Brown, D.H. Ballard, "Model-Driven Vision Using Procedure Description: Motivation and Application to PhotoInterpretation and Medical Diagnosis", SPIE proceedings August 1978

[Michie] Michie, D. "Memo Functions and Machine Learning", Nature, vol. 218, pg. 19-22, 1968